Text-Based Feature-Free Automatic Algorithm Selection

Amanda Salinas-Pinto[1][a], Bryan Alvarado-Ulloa[1][b], Dorit Hochbaum[2][c],
Matías Francia-Carramiñana[1][d], Ricardo Ñanculef[1][e] and Roberto Asín-Achá[1][f]
[1] *Universidad Técnica Federico Santa María, Chile*
[2] *University of California, Berkeley, U.S.A.*
{*amanda.salinas, bryan.alvarado*}*@usm.cl, dhochbaum@berkeley.edu,*

Keywords: Algorithm Selection, Deep Learning, SAT, CSP.

Abstract: Automatic Algorithm Selection involves predicting which solver, among a portfolio, will perform best for a given problem instance. Traditionally, the design of algorithm selectors has relied on domain-specific features crafted by experts. However, an alternative approach involves designing selectors that do not depend on domain-specific features, but receive a raw representation of the problem's instances and automatically learn the characteristics of that particular problem using Deep Learning techniques. Previously, such raw representation was a fixed-sized image, generated from the input text file specifying the instance, which was fed to a Convolutional Neural Network. Here we show that a better approach is to use text-based Deep Learning models that are fed directly with the input text files specifying the instances. Our approach improves on the image-based feature-free models by a significant margin and furthermore matches traditional Machine Learning models based on basic domain-specific features, known to be among the most informative features.

1 INTRODUCTION

Automatic Algorithm Selection (AAS) aims to predict the optimal solver for a given problem instance from a portfolio. Traditionally, this process relies on domain-specific features crafted by experts, which, while effective, limits scalability and transferability due to the need for extensive domain knowledge and labor-intensive analysis.

Recent advances in Deep Learning (DL) (Vaswani et al., 2017), where models learn from raw data, offer a compelling alternative to feature-based models. Previous work (Loreggia et al., 2016) in AAS has transformed raw data into fixed-sized images processed by Convolutional Neural Networks (CNNs), but this still requires image-processing techniques.

Our study introduces a novel text-based deep learning approach that directly processes raw textual files specifying problem instances, simplifying the computational pipeline, and enhancing representation.

In this paper, we present our text-based deep learning framework for AAS and evaluate its performance against traditional image-based and feature-based models. Our analysis shows that text-based models are superior in capturing complex information in problem descriptions, leading to more effective and adaptable algorithm selection strategies as compared to image-based methods. Nevertheless, there is still a gap in performance as compared to specialized feature-base models, and closing this gap will still be the base of future research in the area of feature-free algorithm selection.

Our contributions include demonstrating the feasibility of text-based deep learning for AAS and providing a thorough analysis of how these techniques outperform existing feature-free methods. We establish new benchmarks, advancing the field of feature-free AAS, and offer insights into the performance gap between feature-free and feature-based methodologies.

The subsequent sections review relevant literature, define key terms and criteria, outline our text-based AAS framework, present empirical assessments, and conclude with findings and future research directions.

[a] https://orcid.org/0009-0007-2216-4371
[b] https://orcid.org/0009-0008-7468-5723
[c] https://orcid.org/0000-0002-2498-0512
[d] https://orcid.org/0009-0000-8680-7347
[e] https://orcid.org/0000-0003-3374-0198
[f] https://orcid.org/0000-0002-1820-9019

Salinas-Pinto, A., Alvarado-Ulloa, B., Hochbaum, D., Francia-Carramiñana, M., Ñanculef, R. and Asín-Achá, R.
Text-Based Feature-Free Automatic Algorithm Selection.
DOI: 10.5220/0012913700003838

In *Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2024) - Volume 1: KDIR*, pages 267-274
ISBN: 978-989-758-716-0; ISSN: 2184-3228

2 RELATED WORK

2.1 Algorithm Selection Systems

Automatic Algorithm Selection (AAS), introduced by (Rice, 1976), optimizes computational processes by selecting the most suitable algorithm for a given problem instance. This approach is rooted in the "No Free Lunch" theorem (Adam et al., 2019), which posits that no single algorithm universally excels across all scenarios.

AAS typically employs a training phase to associate problem instance features with algorithm performance. The trained model then evaluates new instances to predict the most effective algorithm. Recent literature has explored AAS in various domains, including timetabling (Seiler et al., 2020; Bossek and Neumann, 2022), SAT (Xu et al., 2008), and Multi-Agent Path-Finding (Bulitko, 2016; Achá et al., 2022).

Kerschke et al. (Kerschke et al., 2019) provide a comprehensive survey of algorithm selection and configuration, introducing a taxonomy that distinguishes between "per-set" and "per-instance" methods. Our focus is on "per-instance" AAS, which considers each problem instance individually.

While many AAS systems employ complex strategies, such as the hybrid methodology of semi-static solver schedules (3S) (Kadioglu et al., 2011) or Autofolio (Lindauer et al., 2015), our study concentrates on straightforward approaches. We assume an ML model receives an instance characterization and selects a single solver to execute until completion or time limit.

Most AAS research relies on domain-specific, expert-crafted features. However, an alternative approach involves developing ML methods that utilize raw/generic instance representations, allowing the learning process to identify relevant features autonomously. This approach was first explored by (Loreggia et al., 2016).

2.2 Deep Learning for Algorithm Portfolios

(Loreggia et al., 2016) introduced a groundbreaking approach to Automatic Algorithm Selection (AAS) based on deep learning. Unlike traditional AAS techniques that use hand-crafted, domain-specific features, this method leverages generic raw data — the text file contents describing the problems.

The process transforms text files into a fixed-size image format suitable for Convolutional Neural Network (CNN) analysis:

1. Convert textual input into a vector of ASCII codes.
2. Reorganize the vector into a $\sqrt{N} \times \sqrt{N}$ matrix, where N is the total character count.
3. Resize the resulting "ASCII image" to a uniform scale.

The CNN can be trained as a multi-class classifier, multi-label classifier, or regressor. Evaluated using SAT and Constraint Satisfaction Problems (CSP) instances, this method showed potential to outperform the Single Best Solver (see Subsection 2.3).

Despite its successes, this approach may not perform as well as methods utilizing domain-specific features.

2.3 Performance Metric for Meta-Solvers

We define an algorithm-selection-based meta-solver as a system comprising a portfolio of solvers. It analyzes an input instance and runs one or more solvers to resolve it. A solver *solves* an instance if it can decide its satisfiability (for decision problems) or find and certify the optimal solution (for optimization problems) within a time limit.

All our meta-solvers here operate uniformly:

1. Accept an input instance.
2. Use an ML model to predict the most efficient solver, identify capable solvers, or estimate solving times.
3. Select and run one solver based on these predictions.

We evaluate the meta-solver's performance using two baselines:

Single Best Solver (SBS): The solver performing best on average across all training instances.

Virtual Best Solver (VBS): A hypothetical meta-solver always choosing the most effective algorithm for each instance.

Performance is measured using the PAR10 metric (Lindauer et al., 2019). For a solver s on instance i:

$$m_s(i) = \begin{cases} t_s(i) & \text{if } t_s(i) \leq \tau \\ 10\tau & \text{otherwise} \end{cases}$$

where τ is the timeout constant and $t_s(i)$ is the solving time.

We use the performance measure $\hat{m}$ (Lindauer et al., 2019) to evaluate meta-solvers:

$$\hat{m}_{ms} = \frac{m_{ms} - m_{VBS}}{m_{SBS} - m_{VBS}} \tag{1}$$

Values of $\hat{m}_{ms}$ close to 0 indicate performance near VBS, while values close to 1 suggest performance similar to SBS. Values above 1 indicate the meta-solver is less effective than SBS.

3 TEXT-BASED FEATURE-FREE AAS

We follow (Loreggia et al., 2016)'s approach, working directly with raw problem instance representations. Our Deep Learning models are fed with raw text representations, rather than pre-processed image-like inputs.

3.1 Architecture Overview

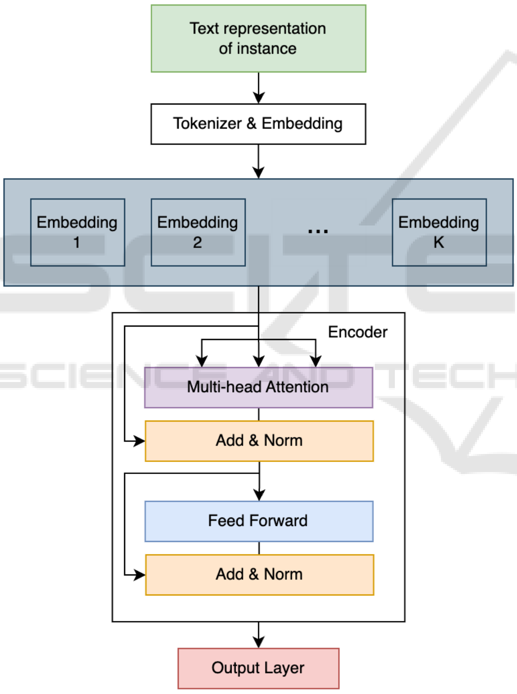

Figure 1: Overall architecture of our text-based Deep Learning Model for AAS.

Our architecture (Figure 1) is a modified Transformer neural network, using only the encoder component similar to the one of (Vaswani et al., 2017). The input text x is truncated, tokenized, and converted into embeddings $\mathbf{x} = \langle \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \rangle$. The encoder's outputs $\mathbf{z} = \langle \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n \rangle$ are fused into a global descriptor $\mathbf{z}$ using Global Max Pooling (Christlein et al., 2019), then mapped to a prediction through a fully connected output layer.

3.2 Tokenizers and Embeddings

We explore two tokenization approaches:

Pre-trained Tokenization: Using *SentencePiece* (Kudo and Richardson, 2018).

Trained Tokenization: Using *Charformer* (Tay et al., 2021).

3.3 Encoder Architecture

Our encoder computes $M = 4$ hierarchical transformations $\mathbf{Z}^{(k)} = \text{EBlock}(\mathbf{Z}^{(k-1)})$. Each block includes a self-attention mechanism and a position-wise feed-forward net. The self-attention mechanism computes:

$$\mathbf{P} = \text{SelfAttention}(\mathbf{Z}) = \text{softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d'}}\right)\mathbf{Z}, \quad (2)$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{Z}$ are learnable matrices that project $\mathbf{Z}^{(k-1)}$ into a d'-dimensional latent space. We use multi-head attention with $H = 4$ heads. The final block's output $\mathbf{Z}^{(k)}$ is obtained after applying a residual connection (He et al., 2016) and layer normalization (Ba et al., 2016) around each sublayer. We did not use positional embeddings.

3.4 Problem Framing Strategies

We explore three strategies:

Multi-Class Classification: Identifies the most suitable solver and the meta-solver runs it. The output layer is a softmax function, and the loss function is categorical cross-entropy.

Multi-Label Classification: Identifies all solvers capable of solving the instance within the defined time limit τ. Each solver corresponds to an element in the output vector, with a sigmoid function applied element-wise. The loss is measured through the Hamming loss function. Since the probabilities here are not complementary, they determine the likelihood that a solver will be fit for the problem instance. The meta-solver executes the solver that exhibits the highest likelihood.

Regression: Estimates normalized log delta runtime for each solver. The mean squared error function serves as the loss function, and the output layer is linear. The meta-solver runs the solver predicted to have the shortest runtime.

$$r_{s,i} = log(1 + m_s(i) - \min_{s \in S}(m_s(i)))$$

$$y_{s,i} = \frac{r_{s,i} - mean(r_{s,i})}{std(r_{s,i})}$$

4 EXPERIMENTAL SETUP AND BASELINES

4.1 Libraries and Hardware

We implemented our Deep Learning models in Python 3.10, using PyTorch 2.0.0. For the text-based models, we used the Charformer tokenizer 0.0.4[1], and SentencePiece 0.2.0. For the image scaling, needed by the image-based models, we used OpenCV 4.7.0.72. The feature-based models were implemented using scikit-learn 1.4.2.

The experiments were carried out on a machine with an Intel Xeon Skylake (2x16 @2.1 GHz) processor and an Nvidia A40 GPU. The machine runs Scientific Linux 7 and has 48GB of RAM.

4.2 Benchmark Sets

To evaluate our approach, first, we aimed to use the same benchmark sets used in (Loreggia et al., 2016). However, the precise sets of instances and the partitions used in that study were not disclosed publicly and could not be provided by the authors when asked in an internal communication. We then searched for similar-nature benchmarks for which the instance files and hand-crafted features used in the AAS community were available. Unfortunately, we could not find meaningful benchmark sets similar to the ones named "SAT Random" and "SAT Crafted" in (Loreggia et al., 2016). However, we were able to collect the most interesting benchmark sets reported in such study, "SAT Industrial" and "CSP". These benchmark sets are the more interesting because of their diversity in size, complexity, and complementarity of the solvers.

SAT Industrial. This benchmark includes instances used in the SAT competition between 2003 and 2016 in the industrial/application categories. The performance of the solvers in these competitions was retrieved from *ASLib*, specifically from the *SAT03-16-INDU-ALGO* scenario. We removed 269 instances that could not be solved by any solver in the portfolio within the given τ time limit. After filtering, the dataset contains $1,730$ instances and 10 different solvers.

CSP. We used the benchmark from the 2009 CSP competition[2]. The performance data for each solver was obtained from the *PROTEUS-2014* scenario (Hurley et al., 2014) in *ASlib*. We filtered the instances by removing the "easy" instances that could be solved by all solvers within the time-limit equivalent to compute the instance's features, in addition to removing the "difficult" instances that were not solved by any of the solvers within the given time limit τ. This resulted in a total of $1,613$ instances and 22 different solvers.

4.3 Data Partitioning and Evaluation Criteria

We split each benchmark into train and test datasets. For the train dataset we used 80% of the total instances, and the remaining 20% is reserved as the test dataset. The training dataset is used for training and model selection, while the test dataset is used to compare the in-production performance of the best text-based, image-based, and feature-based approaches.

To select the best model for each approach, we performed 10-fold cross-validation with the training set. We compared the models based on the $\hat{m}$ metric associated with a meta-solver using them. We then selected the best model based on the mean $\hat{m}$ metric across the different folds.

4.4 Feature-Based Models

To offer a comprehensive view of our study on feature-free models, we also implement and evaluate feature-based models employing both state-of-the-art crafted features and basic informative features, using Random Forest models. The comparison of feature-free models with these feature-based counterparts serves a dual purpose: firstly, to analyze and document the performance disparities between these two paradigms, and secondly, to provide the research community with a benchmark on the effectiveness of applying state-of-the-art crafted features in a straightforward manner on ASLib scenarios that are widely used.

Basic Features: Two basic features extracted from the text describing a problem instance are: the number of variables and the number of constraints. The motivation for these two features is that the instance size usually appears among the most simple and informative ones. We expect that

[1] as implemented in https://github.com/lucidrains/charformer-pytorch

[2] https://www.cril.univ-artois.fr/CSC09/results/globalbybench.php?idev=30&idcat=38&idSubCat=60

training ML models on these two features establishes a baseline for the other methods.

We note here that, for the CSP benchmark, the number of variables and constraints in the text file differ from the *direct_nvariables* and *direct_nclauses* features of the ASLib scenario, since the former seem to be computed after grounding the CSP formula to SAT.

Full Set of Features: These features represent the state-of-the-art in domain-specific algorithm selection, as provided in the corresponding scenarios of *ASLib*. All these 483 SAT features were introduced in (Xu et al., 2008), and constitute the by-default standard for AAS in SAT.

For CSP, ASLib provides the 198 domain-specific features as proposed in (Hurley et al., 2014). We note here that our evaluation does not consider the time needed to compute all these features, even though some of them are expensive to compute and others are captured during runtime, from a reference solver.

Although these features and scenarios are commonly referenced in the literature, we were unable to find reported performance values ($\hat{m}$) for meta-solvers that utilize these features directly. Consequently, our aim is to document these values to serve as a reference for future research.

4.5 Image-Based Models

We implemented the approach presented by (Loreggia et al., 2016) carefully following the experimental setup described there. For the training, we used Stochastic Greedy Descent (SGD) with Nesterov momentum of 0.9 and a learning rate of 0.03. As first layer, we included a batch normalization layer, as proposed in (Ioffe and Szegedy, 2015). The output layer changes depending on the learning task, as mentioned in Section 3.4. We set a training batch size of 128 and 100 epochs.

4.6 Text-Based Models

Due to limitations on the hardware needed to train our model on arbitrary-size instances, we truncated the size of the instances to $10,000$ characters. To avoid introducing biases into the model, we removed from the text files any comments or other kind of meta-information like the folder name where the instance is located, or the name of the generator of the instance. In a preliminary evaluation, we noted that these meta-information fields may unfairly help the text-based models and decided not to consider this information.

For training our text-based models, we used AdamW optimizer (Loshchilov and Hutter, 2017) with a *learning rate* of 10^{-5}. We set a batch size of 8 samples and an *embedding size d* of 128. The training was set to take 100 *epochs*.

Since the sequence length produced by SentencePiece can vary among instances while our encoder accepts a fixed-length sequence, we computed the median length of SentencePiece's output, truncating longer sequences and padding shorter ones. The vocabulary size *v* for SentencePiece, was set to 1024. Charformer, which operates at the character level, had a vocabulary size of 257 (256 ASCII values plus one token reserved for padding). We set the *max block size* and the *downsample factor* to their default values (4). Additionally, we employed the block attention scores proposed in Section 2.1.4 of (Tay et al., 2021) to form latent subwords.

5 RESULTS

5.1 Feature-Based Validation Results

Table 1: $\hat{m}$ metric values across 10-fold validation sets for different handcrafted-features-based meta-solvers for CSP and SAT Industrial benchmark sets. Here, HF = Handcrafted-based, F=Full set of features, B=Basic set of features, ML=Multi-label model, Reg= Regression model, MC=Multi-class model.

Model	CSP	SAT Industrial
HF-F-ML	**0.409 ± 0.064**	0.680 ± 0.312
HF-F-Reg	0.416 ± 0.087	**0.640 ± 0.291**
HF-F-MC	0.546 ± 0.066	0.676 ± 0.228
HF-B-ML	0.638 ± 0.068	1.054 ± 0.361
HF-B-Reg	**0.557 ± 0.066**	**0.939 ± 0.365**
HF-B-MC	0.582 ± 0.075	1.22 ± 0.387

Table 1 shows the average and standard deviation of the $\hat{m}$ values computed across 10-fold cross-validation subsets for six feature-based meta-solvers. The first three meta-solvers are based on the full set of features provided in the ASLib, while the last three meta-solvers only use the two basic features related to the size of the instances. For a fair comparison with our feature-free model, these feature-based meta-solvers can cast AAS as a multi-label task (ML), a regression task (Reg), or a multi-class (ML) problem. As can be seen, for the CSP benchmark, the most successful meta-solver using the full set of features is the one based on multi-label classification (ML). In contrast, for the SAT Industrial benchmark, the best meta-solver, using the full set of features, is the one based on regression (Reg). Nevertheless, we note that even for these state-of-the-art crafted features, the meta-solvers are quite sensible to the test set in SAT, as is evident from the considerable standard deviation.

Regarding the meta-solvers using only the two basic features, the meta-solvers based on regression show better performance in both benchmark sets. We note that, on average, only using these two basic features allows the meta-solvers to outperform the SBS. For CSP, we found a considerable margin of advantage, and for SAT Industrial, a smaller margin.

We report the performance of these feature-based solvers in the test set in Subsection 5.3. All the results reported here are consistent with the literature.

5.2 Image-Based Validation Results

Table 2: $\hat{m}$ metric values across 10-fold validation sets for different image-based meta-solvers for CSP and SAT Industrial benchmark sets. Here, Im = Image-based, ML=Multi-label model, Reg= Regression model, MC=Multi-class model.

Model	CSP	SAT Industrial
Im-ML	0.640 ± 0.088	1.25 ± 0.407
Im-Reg	**0.609 ± 0.104**	**1.14 ± 0.346**
Im-MC	0.898 ± 0.109	1.66 ± 0.527

Table 2 shows the statistics of the $\hat{m}$ values computed across 10-fold cross-validation subsets for three image-based meta-solvers. For a fair comparison with our text-based model, we trained image-based meta-solvers based on multi-label, regression, and multi-class formulations. The results in Table 2 demonstrate that, although the regression approach was not considered in (Loreggia et al., 2016), the most successful image-based meta-solver is the one based on regression for both benchmark sets.

The meta-solver for CSP outperforms CSP's Single Best Solver by a significant margin while maintaining a considerable gap with the Virtual Best Solver for CSP. These results are in line with the ones reported in (Loreggia et al., 2016). However, an exact match between our image-based results and those in (Loreggia et al., 2016) is virtually impossible since the training/validation/test differ.

Image-based SAT meta-solvers cannot outperform the Single Best Solver. This result diverges from the results of (Loreggia et al., 2016), which reported an image-based meta-solver that outperforms SBS on SAT. This discrepancy may happen due to differences in the specific SAT industrial benchmark set used or differences in the training/test partitions. However, we also observe that the performance of the SAT image-based meta-solver varies significantly depending on the training and validation set (standard deviation of 0.346 among cross-validation folds).

5.3 Text-Based Validation Results

Table 3: $\hat{m}$ metric values across 10-fold validation sets for different text-based ML models for CSP and SAT Industrial benchmark sets. Here, Txt = Text-based, Cha=Trained tokenizer Charformer, Sen= Pre-trained tokenizer Sentenpiece, ML=Multi-label model, Reg= Regression model, MC=Multi-class model.

Model	CSP	SAT Industrial
Txt-Cha-ML	0.488 ± 0.047	0.952 ± 0.281
Txt-Cha-Reg	**0.469 ± 0.050**	**0.889 ± 0.303**
Txt-Cha-MC	0.581 ± 0.076	1.312 ± 0.354
Txt-Sen-ML	0.482 ± 0.082	1.078 ± 0.252
Txt-Sen-Reg	0.536 ± 0.100	1.119 ± 0.448
Txt-Sen-MC	0.608 ± 0.120	1.470 ± 0.319

Table 3 shows the average and standard deviation of the $\hat{m}$ values for our text-based meta-solvers computed by 10-fold cross-validation. The first three meta-solvers are text-based models jointly trained with the tokenizer (Charformer), while the last three meta-solvers use the pre-trained tokenizer (SentencePiece). As can be seen, the most successful meta-solver is the one that uses a regression model jointly trained with the tokenizer.

The CSP meta-solver significantly improves the performance of the SBS for this domain. With an average $\hat{m}$ value equal to 0.469 and little standard deviation, this meta-solver's performance can be interpreted as closer to the VBS than to the SBS.

Despite the formulation, obtaining a $\hat{m}$ lower than 1 for SAT Industrial was impossible using image-based methods. Noticeably, our best text-based meta-solver outperforms the Single Best Solver with an average $\hat{m}$ value of 0.889 in this benchmark. Nevertheless, as for the previous models, the standard deviation is high (0.303), which suggests that the meta-solver's performance varies considerably depending on the validation instances used.

5.4 Test Set Results

Here we compare feature-based, image-based and text-based meta-solvers on the *test set* of each benchmark. For each category, we selected the best approach using 10-fold cross-validation, and trained the model with the whole training set. Again, we note that results given on feature-based models are reported as a reference to gain perspectives as well as to communicate performance values of meta-solvers using straightforward models.

As anticipated, the meta-solvers that yield the best

Table 4: $\hat{m}$ metric values of the testing set, for each "best" model for each approach and benchmark set.

Model	CSP	SAT Industrial
HF-B-Reg	0.549	0.975
HF-F-ML	**0.442**	—
HF-F-Reg	—	**0.674**
Im-Reg	0.642	1.309
Txt-Char-Reg	**0.556**	**1.037**

results are those that utilize expert-designed features specific to the domain. In the case of CSP, the meta-solver employing a multi-label classification model achieves an $\hat{m}$ value of 0.442. This significantly narrows the performance disparity between the SBS and the VBS in CSP scenarios. Similarly, for the SAT Industrial benchmark, the regression-based meta-solver records an $\hat{m}$ value of 0.674. Considering the complexity of this benchmark, this score is notably satisfactory. These outcomes align with those from contemporary meta-solvers specialized for CSP and SAT Industrial. It is important to note that this assessment only gauges the effectiveness of the features in a well-adjusted ML model. This overview omits the consideration that many sophisticated features, while beneficial, are computationally intensive and may not be regularly employed in elaborate Algorithm Selection Systems that utilize both a presolver and a solver scheduler. Hence, the current $\hat{m}$ values of the meta-solvers that incorporate these advanced features likely represent a lower bound for any straightforward methodology.

When comparing the two feature-free meta-solvers, our text-based method significantly surpasses the image-based method and nearly matches the performance of the meta-solvers that incorporate the two basic crafted features. This suggests that the image-based models may fail to capture even basic information, such as the size of the problem instance. Conversely, the text-based models appear capable of recognizing information akin to these features, even though our system uses only basic vanilla encoders. Converting these $\hat{m}$ scores to average running times reveals that the expected average time for the text-based model is approximately 13% lower than that of the image-based model for the CSP benchmark. For the SAT Industrial benchmark, this reduction is about 20%. Collectively, these figures demonstrate that our novel text-based feature-free framework significantly decreases the performance gap between feature-free and feature-based Algorithm Selection Systems (AAS).

6 CONCLUSIONS AND FUTURE WORK

We present here a novel approach to Automatic Algorithm Selection that leverages the capabilities of text-based deep learning models. Our results clearly demonstrate that this method not only simplifies the feature extraction process (by eliminating the need of image-based preprocessing) but also significantly enhances the performance of existing feature-free algorithm selection paradigms. By directly processing raw textual descriptions of problem instances, our approach has shown a marked improvement over traditional, image-based CNN approaches in terms of both performance and robustness across benchmarks.

The effectiveness of our method was validated through extensive experiments on benchmarks containing a variety of problem instances. The experimental results underscore the potential of deep learning techniques that operate directly on raw data, providing a more scalable and flexible end-to-end solution for the field of AAS.

Our experiments confirm that, up to date, no feature-free algorithm selection approach can outperform meta-solvers based on validated domain-specific crafted features by experts. However, results also show that text-based feature-free models can match the performance of meta-solvers based on basic informative features. This finding suggests that deep learning methods can learn problem representations beyond the most crude and elementary characterization.

While our study has made significant strides in the application of text-based models to algorithm selection, several avenues remain open for further exploration. Future work may include:

- **More Complex AAS Systems:** Our proposal can be the base for more complex AAS systems, including dynamic portfolios and schedulers.
- **More Complex ML Models:** More complex transformer architectures can also be tested. Besides, AAS can be framed in a more sophisticated way to leverage advances in ranking, metric learning, and recommender systems.
- **Handling the Whole Text Files:** A plethora of architectures have been proposed for long text modeling in deep learning. These methods should be systematically evaluated to overcome the limitations of our text-based meta-solver.
- **Anytime AAS:** Extending our method to Anytime Algorithm Selection could significantly benefit environments where decisions should be made based on the available computational resources.

- **Transfer Learning:** Exploring transfer learning techniques to adapt models trained on one set of problem instances to handle others effectively could contribute to a general purpose AAS.
- **Interpretable AI Models:** Enhancing the interpretability of deep learning models used in AAS to provide insights into why certain algorithms are preferred for specific instances could help refine the models further and in gaining trust from users.
- **Benchmarks and Datasets:** Applying our framework to other domains, possibly including optimization problems whose domain metrics $\hat{m}$ involve the values of the objective function.

In conclusion, the research presented in this paper sets a new benchmark in the field of feature-free AAS and opens up numerous possibilities for the evolution of more intelligent and autonomous algorithm selection systems. Our future efforts will focus on expanding the capabilities of our framework and exploring these promising directions to further enhance the field of algorithm selection.

ACKNOWLEDGEMENTS

Authors 1st, 2nd, 3rd, 4th, and 6th are supported in part by AI institute NSF award 2112533.

REFERENCES

Achá, R. A., López, R., Hagedorn, S., and Baier, J. A. (2022). Multi-agent path finding: A new boolean encoding. *Journal of Artificial Intelligence Research*, 75:323–350.

Adam, S. P., Alexandropoulos, S.-A. N., Pardalos, P. M., and Vrahatis, M. N. (2019). No free lunch theorem: A review. *Approximation and optimization: Algorithms, complexity and applications*, pages 57–82.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bossek, J. and Neumann, F. (2022). Exploring the feature space of tsp instances using quality diversity. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 186–194.

Bulitko, V. (2016). Evolving real-time heuristic search algorithms. In *Artificial Life Conference Proceedings 13*, pages 108–115. MIT Press.

Christlein, V., Spranger, L., Seuret, M., Nicolaou, A., Král, P., and Maier, A. (2019). Deep generalized max pooling. In *2019 International conference on document analysis and recognition (ICDAR)*, pages 1090–1096. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hurley, B., Kotthoff, L., Malitsky, Y., and O'Sullivan, B. (2014). Proteus: A hierarchical portfolio of solvers and transformations. In *Integration of AI and OR Techniques in Constraint Programming: 11th International Conference, CPAIOR 2014, Cork, Ireland, May 19-23, 2014. Proceedings 11*, pages 301–317. Springer.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr.

Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., and Sellmann, M. (2011). Algorithm selection and scheduling. In *International Conference on Principles and Practice of Constraint Programming*, pages 454–469. Springer.

Kerschke, P., Hoos, H. H., Neumann, F., and Trautmann, H. (2019). Automated algorithm selection: Survey and perspectives. *Evolutionary computation*, 27(1):3–45.

Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Lindauer, M., Hoos, H. H., Hutter, F., and Schaub, T. (2015). Autofolio: An automatically configured algorithm selector. *Journal of Artificial Intelligence Research*, 53:745–778.

Lindauer, M., van Rijn, J. N., and Kotthoff, L. (2019). The algorithm selection competitions 2015 and 2017. *Artificial Intelligence*, 272:86–100.

Loreggia, A., Malitsky, Y., Samulowitz, H., and Saraswat, V. (2016). Deep learning for algorithm portfolios. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Rice, J. R. (1976). The algorithm selection problem. In *Advances in computers*, volume 15, pages 65–118. Elsevier.

Seiler, M., Pohl, J., Bossek, J., Kerschke, P., and Trautmann, H. (2020). Deep learning as a competitive feature-free approach for automated algorithm selection on the traveling salesperson problem. In *International Conference on Parallel Problem Solving from Nature*, pages 48–64. Springer.

Tay, Y., Tran, V. Q., Ruder, S., Gupta, J., Chung, H. W., Bahri, D., Qin, Z., Baumgartner, S., Yu, C., and Metzler, D. (2021). Charformer: Fast character transformers via gradient-based subword tokenization. *arXiv preprint arXiv:2106.12672*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Xu, L., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2008). Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, 32:565–606.